Internship Report

FEB 2021-JUNE 2021

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Aman Choubey (181338)

to

Department of Computer Science & Engineering and Information Technology **Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh**

Candidate’s Declaration

I hereby declare that the work presented in this report entitled **“ Internship report.”** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2015 to December 2015 under the supervision of **Mr. Bobbin Sondhi**.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

[illegible]

Aman Choubey

(181338)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

[illegible]

Mr. Bobbin Sondhi

Senior Software Engineer

May 27th 2022

Acknowledgement

I take this opportunity to express my sincere thanks and deep gratitude to all those people who extended their wholehearted cooperation and have helped me in completing this internship successfully.

Foremost, I would like to thank Mr. Ritesh Arora, who mentored me, guided me and challenged me.

I also thank my family and friends who greatly supported me during the course of the Internship.

Last but not the least, I would like to thank our founders for considering me a part of the organization and provide such a great Platform to learn and enhance my skills

Aman Chouybey

181338

Jaypee University of Information Technology

Table of content

Candidate's Declaration	**3**
Acknowledgement	**4**
Table of content	**5**
Chapter 1-Introduction	**7**
Paxcom	7
Paymentus	9
How Payments Works:	11
Chapter 2 -Tools and Technologies	**12**
JavaScript	12
Node JS	13
How node JS works ?	15
Express JS	17
Why Express JS?4	18
The features of node JS help express saving a lot of time.	18
Mongo DB	19
Why Use MongoDB?	20
Chapter 3 - Natours	**22**
MVC architecture	22
Advantage of MVC	25
Rest API	26
Documentation	28
Authorization	46
What is JWT?	46
Login and reset password	46
Unit testing	48
Unit testing with JS	48

How to unit test rest API 48

Tools for unit testing in JS 50

Mocha 50

Chai 50

Chai http 50

Chapter 4- Front End 51

HTML 51

Javascript 52

Typescript 53

Chapter 5- Result and Conclusion 56

References 57

Chapter 1-Introduction

Paxcom

Figure 1: Paxcom Logo

We.are.a.team.of.200+Ecommerce.enthusiasts,.passionate.about.using.technology.to.simplify. Digital Commerce .and .Payments .for .brands .across .the .globe. .Paxcom .is .a .part .of .the .Paymentus .group .- .a leading.global.paperless.electronic.billing.and.payment.

solution.provider.

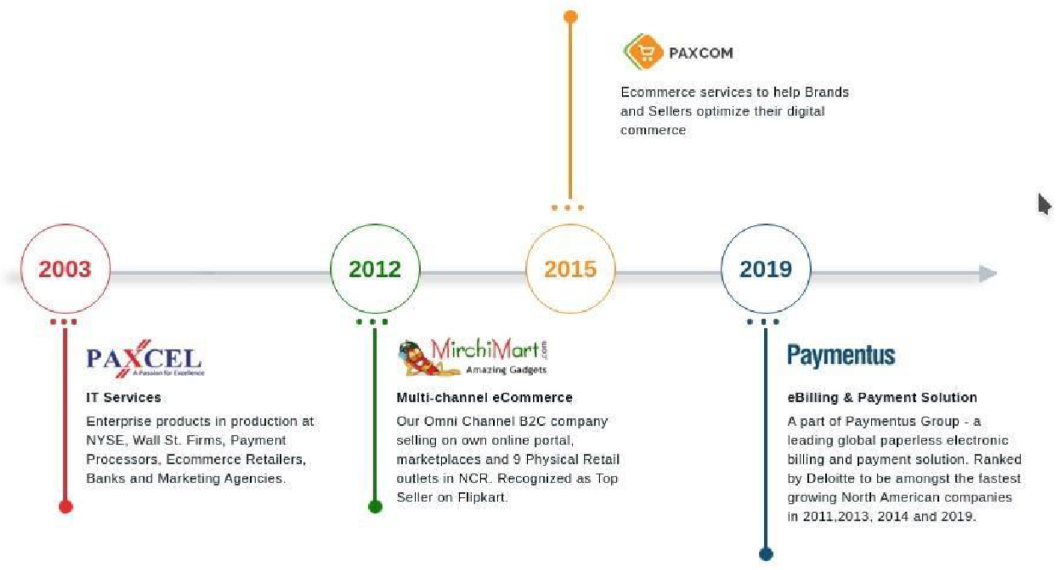

PAXCOM
Ecommerce services to help Brands and Sellers optimize their digital commerce
2003
2012
2015
2019
PAXCEL
A Passion for Excellence
IT Services
Enterprise products in production at NYSE, Wall St. Firms, Payment Processors, Ecommerce Retailers, Banks and Marketing Agencies.
MirchiMart.com
Amazing Gadgets
Multi-channel eCommerce
Our Omni Channel B2C company selling on own online portal, marketplaces and 9 Physical Retail outlets in NCR. Recognized as Top Seller on Flipkart.
Paymentus
eBilling & Payment Solution
A part of Paymentus Group - a leading global paperless electronic billing and payment solution. Ranked by Deloitte to be amongst the fastest growing North American companies in 2011,2013, 2014 and 2019.

Figure 2: Paxcom Timeline

Paxcom characterized way empowers the making of a uniform code base over an association, without agonizing over characterizing certain principles.

Angular as of now accompanies these principles out of the crate. Tailing them doesn't just build the consistency and hence the nature of the code. It makes your application more obvious, as well. That is, in the event that you are as of now acquainted with angular. This exacting methodology likewise proves to be useful across collaboration fringes. It empowers new engineers to incorporate into another group rapidly, as a result of the high commonality with the code.

What I need to state is, you should follow the angular structure rules to capitalize on the angular system. It will make your life significantly simpler, when coming into new undertakings and will expand the nature of your code consequently.

A typical activity, when learning angular is to put everything into the application module. Obviously, that your application will advance into a total chaos. Modules are there which is as it should be!

Modules help to sort out your code into littler groups to make discovering things simpler. Yet, they are not just a corrective thing. With the assistance of lethargic stacked modules, you can likewise build the client experience by just downloading the pieces of the application, that are required at that point.

Angular is written in typescript by Google development team and is maintained by them regularly. It's first initial release was in beginning of 2016. Very first version was called as Angular-JS so in future updates to make it distinguish from other they named it to just Angular and removed the JS. 11 stable versions has been released by the google team along with the community support as of date 11 Nov 2020. Todo material design in anular, it provides angular material library.

Paymentus

Paymentus is a North Carolina based software company providing complete billing solutions.

I worked at Paxcom India Pvt Ltd at Gurugram branch. I worked with the OmniChannel Team and was successfully able to understand their working structure and pattern. I also learned soft skills like communicating within a corporate firm and working with a team. I was successfully able to understand various coding paradigms used by a company to build its application. I got a thorough understanding of some of the company's existing products and some of the upcoming products.

The working culture of the company is great. I thoroughly enjoyed myself working there. Paxcom has atypical blend of work and fun. Although I didn't get to spend much time in the company office and started working from home after the coronavirus pandemic lockdown, I really enjoyed the weekend football and various parties at the company. And even after the lockdown the communication between me and my team was good through various meeting platforms like skype and google meets.

There are numerous things I love about my internship - the experience/information I've acquired, the balance between fun and serious activities climate, my director and colleagues, and surprisingly the gathering of people l exercise with on my mid-day breaks. In any case, in the event that I needed to put it on a certain something, what I like most is, at last, the way that this chance truly opened the entryway into my profession. The most stunning thing about this temporary position experience is that it truly overcame any barrier between learning things and really applying a portion of this information into a reality, and getting paid for it! The way that I presently have an entry-level position straightforwardly identified with what I concentrate on it makes me study more diligently and makes me need to sort out how I can apply a greater amount of what I'm realizing in an internship. It's interesting, in light of the fact that things I've learned in college help me at work.

1.3 Description of Industry:

InT2004,TPaymentusTwasTbornTfromTaTdesireTtoTimproveTtheTwayTbillsTgetTpaid. Vision,TinnovationTandTexemplaryTserviceThaveTpropelledTPaymentusTtoTbecomeT

theTleadingTpaperlessTelectronicTbillingTandTpaymentTsolutionTonTtheTmarket,T

resultingTinT1,300TclientsTincludingTsomeTofTtheTlargestTbillersTinTNorthAmerica.

WeTknowTthatTinTorderTtoTkeepTourTsolutionsTcurrentTandTrelevant,TweTneed peopleTwithTtheTknow how,TdriveTandTproclivityTforTfosteringTaTsupremelyT

Happy customerTexperience.TOurThighlyTcommitted,TcreativeTemployeesTturnedTan

Idea into aTsecure,TSAAS basedTCustomerTEngagementTandTPaymentTPlatform;T

oneTthat enablesTdirect billTorganizationsTtoTprovideTaTunified customer experienceT

andTboostTadoptionTofTcost-savingTelectronicTbillingTandTpaymentTservices.

RecognizedTbyTDeloitteTtoTbeTamongTtheTfastestTgrowingTNorthAmericanT

companiesTinT2011,T2013,T2014,TandT2016,TPaymentusTconsistentlyTstrivesTtoT

developTbetter,Tfaster,TmoreTsecure,TcostefficientTbillingTandTpaymentTplatforms.TWeT continuallyTseekThigherTvalueTforTourTcustomers,TinTbothTsolutionsTandT

service. It'sTwhatThasTledTtoTourTremarkableTgrowthTinTtheTlastTdecade.TWeT succeed whenTourTclientsTsucceed.TTheyTsucceedTwhenTtheirTcustomerT

relationshipsTareTenhancedTand,TinTturn,TtheirTcustomersTparticipateTinTtheseTcostshavingTelectronicTservicesTatThighTrates.

How Payments Works:

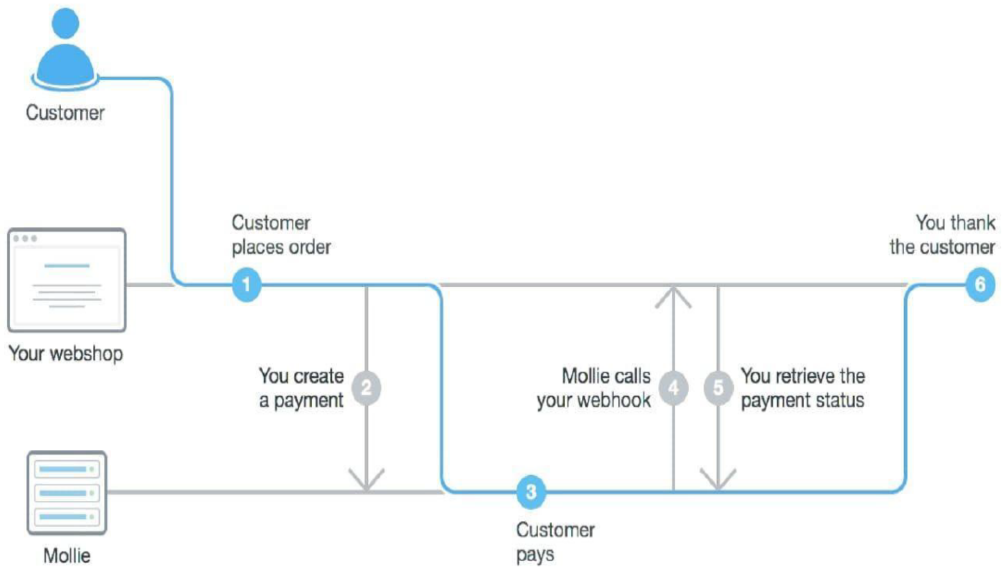


Figure 3: How paymentus works

How to Pay:

Input the URL of the organization you want to pay into your web browser and navigate to the billing section to begin the payment process.

Unsure of the URL?:

It is often found on your paper billing statement. If not, look-up the webpage via your preferred search engine (Google, Bing, etc.).

Chapter 2 -Tools and Technologies

JavaScript

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles. Read more about JavaScript.

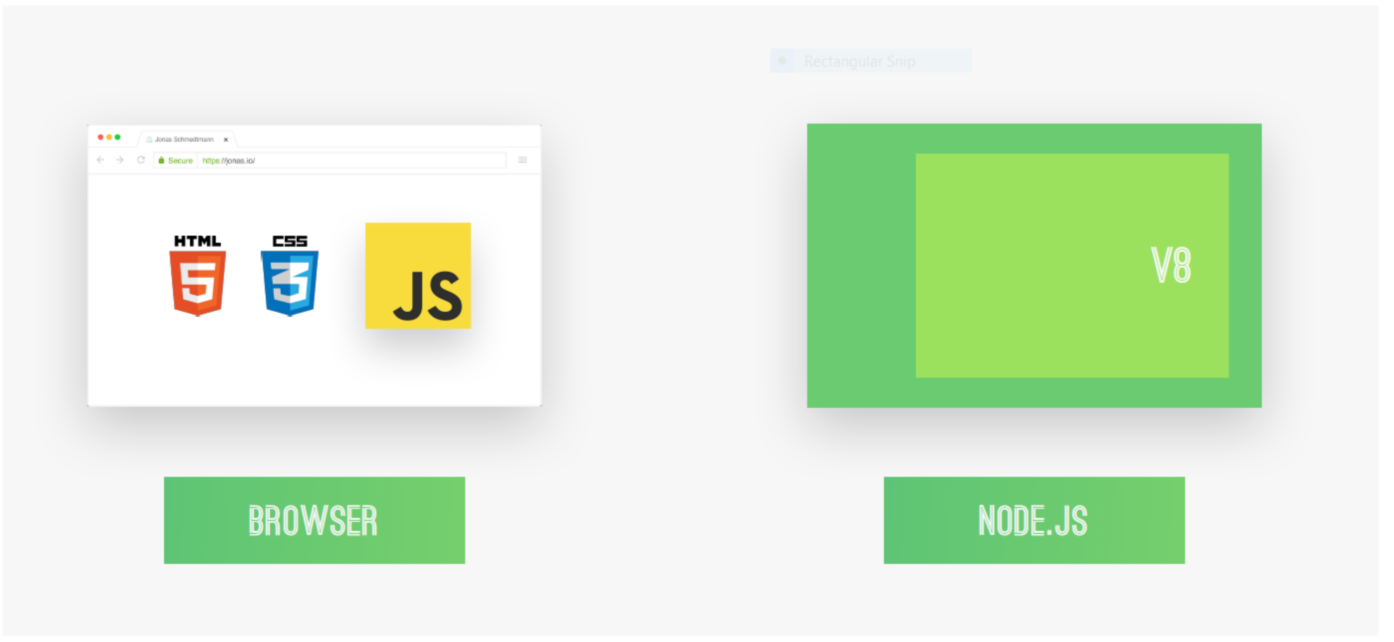
JavaScript is a scripting or programming language that allows you to implement complex features on web pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. — you can bet that JavaScript is probably involved. It is the third layer of the layer cake of standard web technologies, two of which (HTML and CSS) we have covered in much more detail in other parts of the Learning Area.

- HTML is the markup language that we use to structure and give meaning to our web content, for example defining paragraphs, headings, and data tables, or embedding images and videos in the page.

- CSS is a language of style rules that we use to apply styling to our HTML content, for example setting background colors and fonts, and laying out our content in multiple columns.

- JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else. (Okay, not everything, but it is amazing what you can achieve with a few lines of JavaScript code.)

Node JS



Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of project!

Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside the browser. This allows Node.js to be very performant.

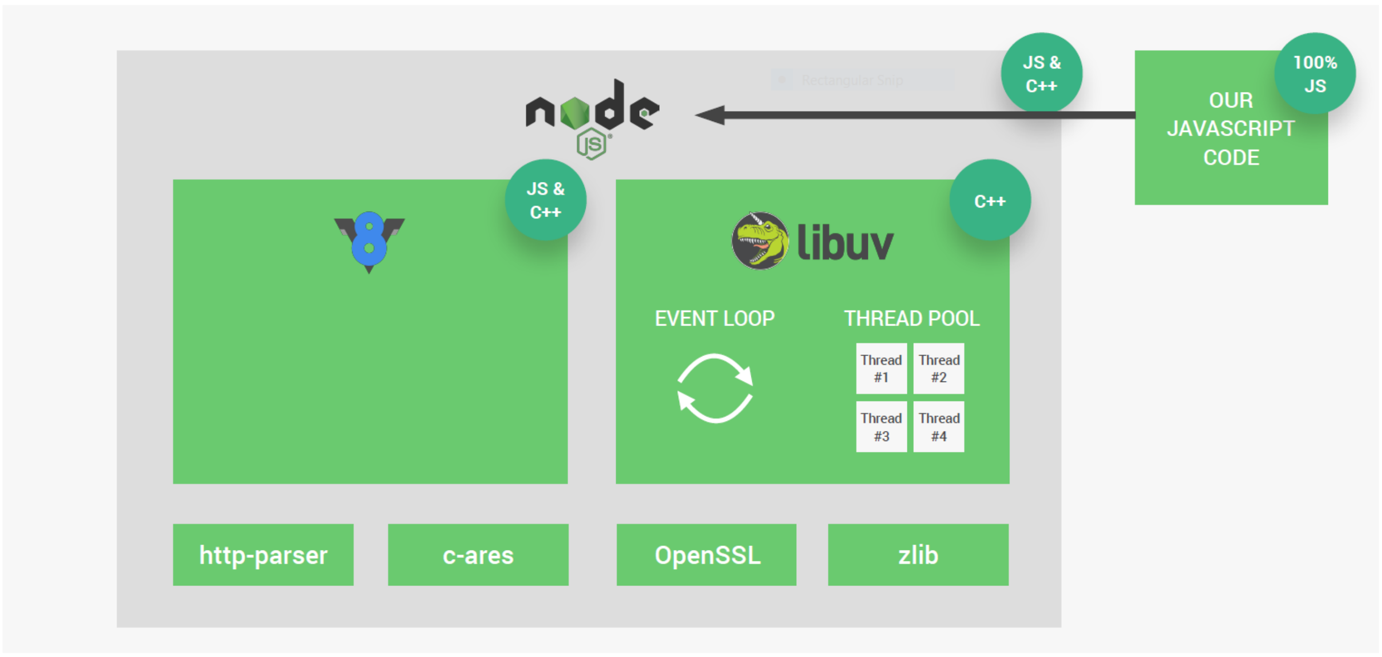
A Node.js app runs in a single process, without creating a new thread for every request. Node.js provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking and generally, libraries in Node.js are written using non-blocking paradigms, making blocking behaviour the exception rather than the norm.

When Node.js performs an I/O operation, like reading from the network, accessing a database or the filesystem, instead of blocking the thread and wasting CPU cycles waiting, Node.js will resume the operations when the response comes back.

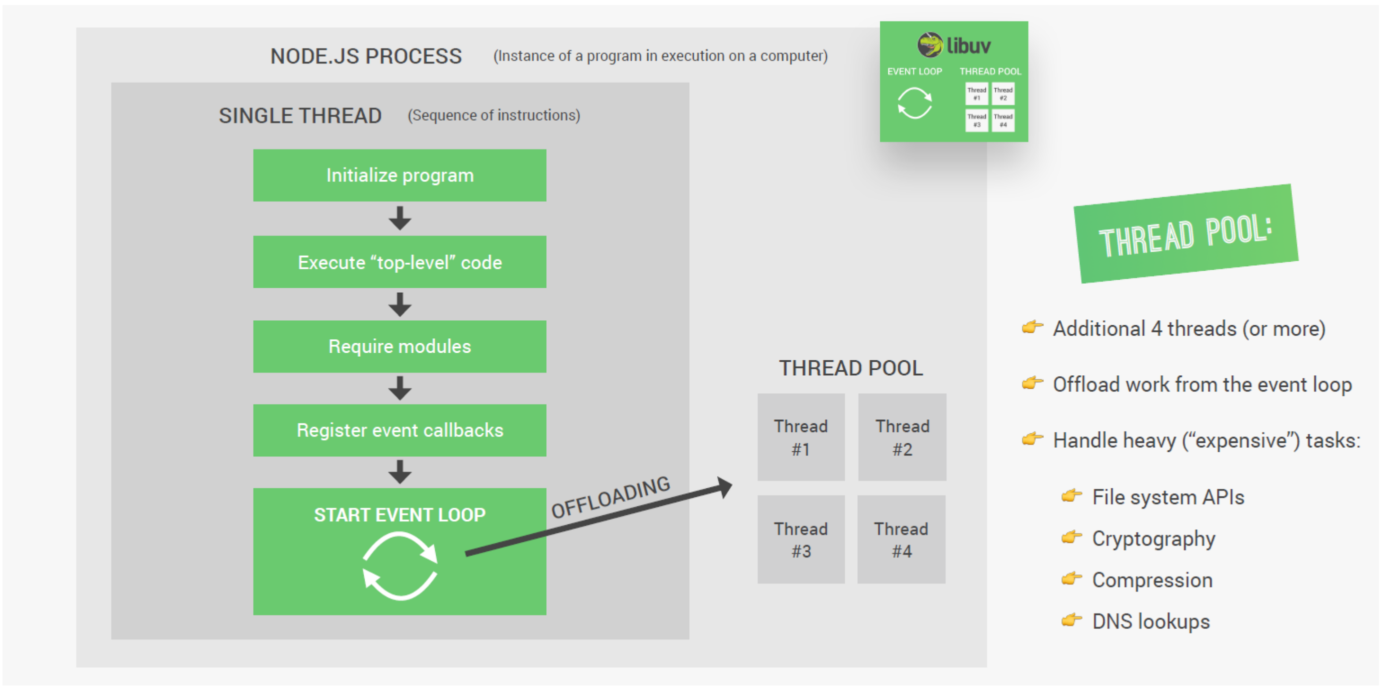
This allows Node.js to handle thousands of concurrent connections with a single server without introducing the burden of managing thread concurrency, which could be a significant source of bugs.

Node.js has a unique advantage because millions of frontend developers that write JavaScript for the browser are now able to write the server-side code in addition to the client-side code without the need to learn a completely different language.

How node JS works ?

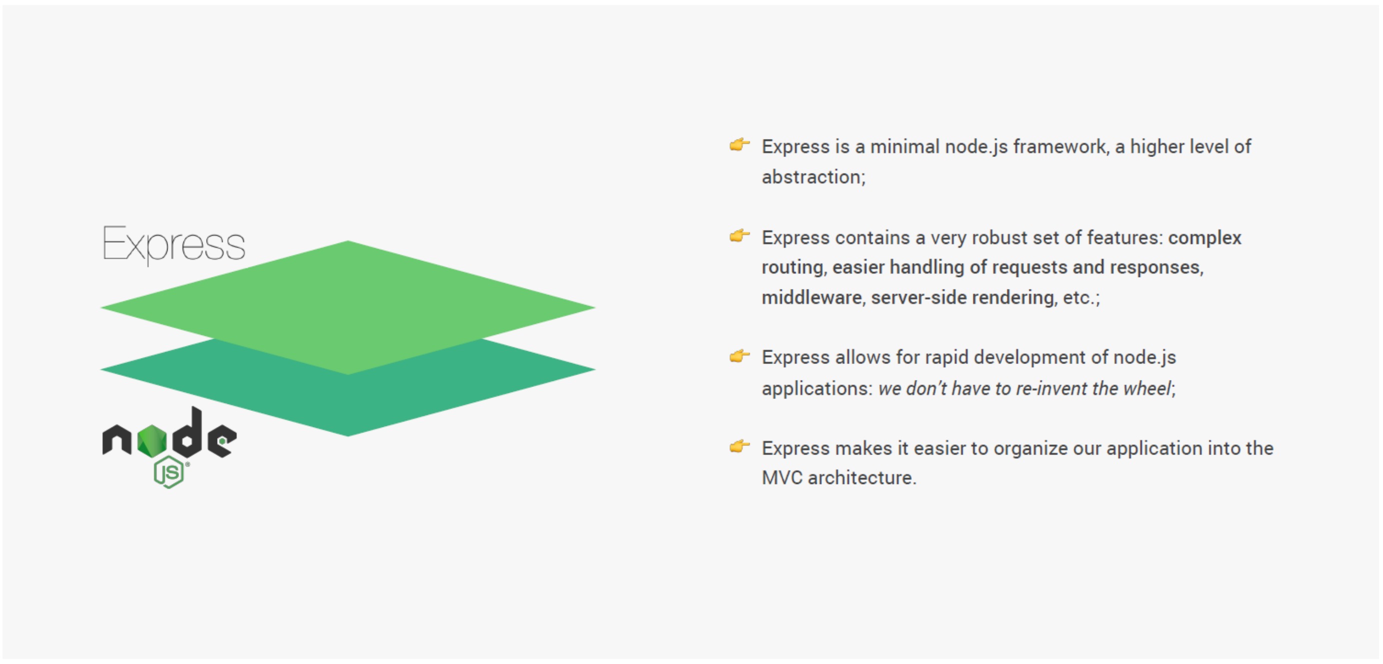


In normal synchronous (threaded) programming, you call a function which does something while you wait. When the function returns, the action is finished and the data is available as the return value. In the asynchronous model, functions always return immediately, or as close to immediately as possible. Any operation that would require waiting (for example, reading from a socket, connecting to a database, executing a query) is done by passing as a parameter a callback function that should be run when that action is done. The action is begun in the background and the function returns immediately. At some point in the future -- a point outside your control -- the OS will signal that the action is complete and the framework will call the callback. It's usually passed some kind of parameter that contains all the things relevant to servicing this request (i.e. a context object) and from that it figures out where to pick up and continue with the overall processing of the work. The programmer explicitly breaks up the overall work into these small steps that are punctuated by waiting for something else.



Express JS

Express is a node JS web application framework that provides broad features for building web and mobile applications. It is used to build a single page, multipage, and hybrid web application.It's a layer built on the top of the Node JS that helps manage servers and routes.


Express
node JS

- Express is a minimal node.js framework, a higher level of abstraction;
- Express contains a very robust set of features: complex routing, easier handling of requests and responses, middleware, server-side rendering, etc.;
- Express allows for rapid development of node.js applications: *we don't have to re-invent the wheel*;
- Express makes it easier to organize our application into the MVC architecture.

Why Express JS?4

- Express was created to make APIs and web applications with ease,
- It saves a lot of coding time almost by half and still makes web and
- mobile applications are efficient.
- Another reason for using express is that it is written in JavaScript, as JavaScript is an easy language even if you don't have a previous
- knowledge of any language. Express lets so many new developers enter the field of web development.

The reason behind creating an express framework for node JS is:

- Time-efficient
- Fast
- Economical
- Easy to learn
- Asynchronous
- Features of Express JS
- Fast Server-Side Development

The features of node JS help express saving a lot of time.

- Middleware

  Middleware is a request handler that has access to the application's request-response cycle.

- Routing

  It refers to how an application's endpoint's URLs respond to client requests.

- Templating

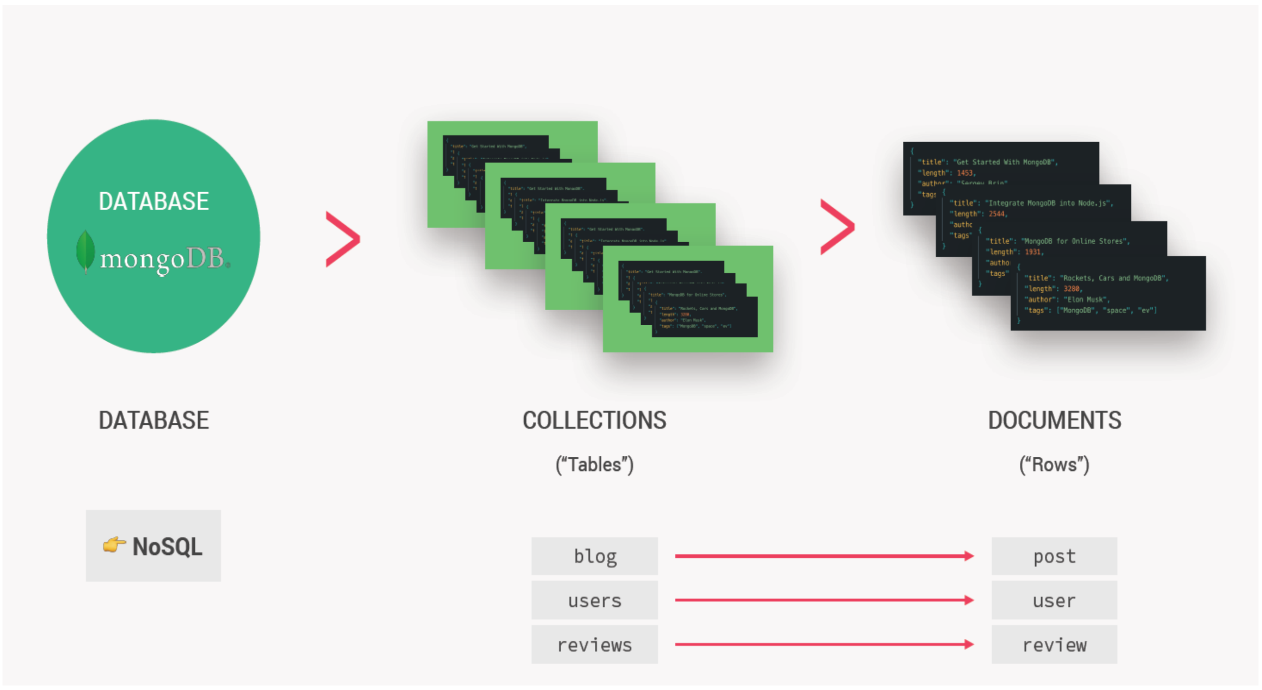
  It provides templating engines to build dynamic content on the web pages by creating HTML templates on the server.

- Debugging

  Express makes it easier as it identifies the exact part where bugs are.

Mongo DB

MongoDB is a document database used to build highly available and scalable internet applications. With its flexible schema approach, it’s popular with development teams using agile methodologies. Offering drivers for all major programming languages, MongoDB allows you to immediately start building your application without spending time configuring a database.

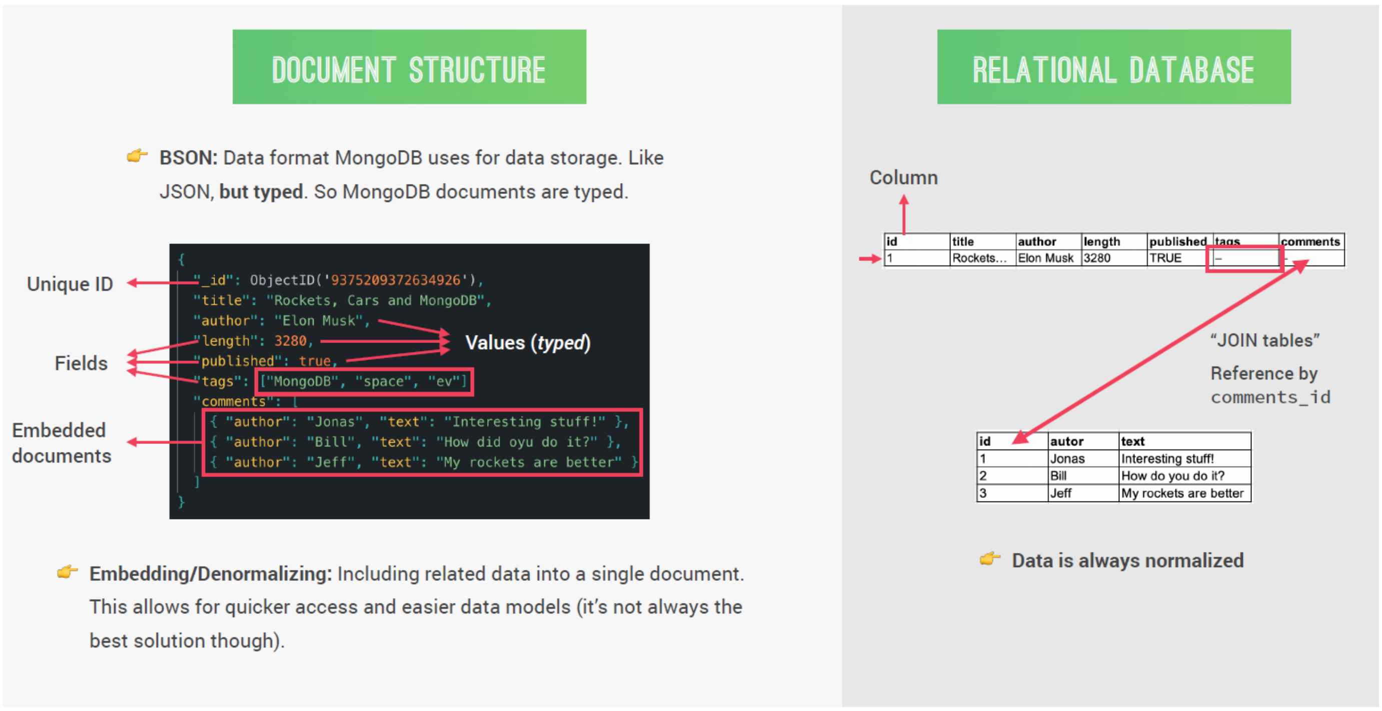


Why Use MongoDB?

MongoDB is built on a scale-out architecture that has become popular with developers of all kinds for developing scalable applications with evolving data schemas.

As a document database, MongoDB makes it easy for developers to store structured or unstructured data. It uses a JSON-like format to store documents. This format directly maps to native objects in most modern programming languages, making it a natural choice for developers, as they don't need to think about normalizing data. MongoDB can also handle high volume and can scale both vertically or horizontally to accommodate large data loads.

MongoDB was built for people building internet and business applications who need to evolve quickly and scale elegantly. Companies and development teams of all sizes use MongoDB for a wide variety of reasons.



BSON: Data format MongoDB uses for data storage. Like JSON, **but typed**. So MongoDB documents are typed.

```
{
  "_id": ObjectID('9375209372634926'),
  "title": "Rockets, Cars and MongoDB",
  "author": "Elon Musk",
  "length": 3280,
  "published": true,
  "tags": ["MongoDB", "space", "ev"]
  "comments": [
    { "author": "Jonas", "text": "Interesting stuff!" },
    { "author": "Bill", "text": "How did oyu do it?" },
    { "author": "Jeff", "text": "My rockets are better" }
  ]
}
```

Embedding/Denormalizing: Including related data into a single document. This allows for quicker access and easier data models (it's not always the best solution though).

id	title	author	length	published	tags	comments
1	Rockets...	Elon Musk	3280	TRUE	–	-

id	autor	text
1	Jonas	Interesting stuff!
2	Bill	How do you do it?
3	Jeff	My rockets are better

Data is always normalized

Chapter 3 - Natours

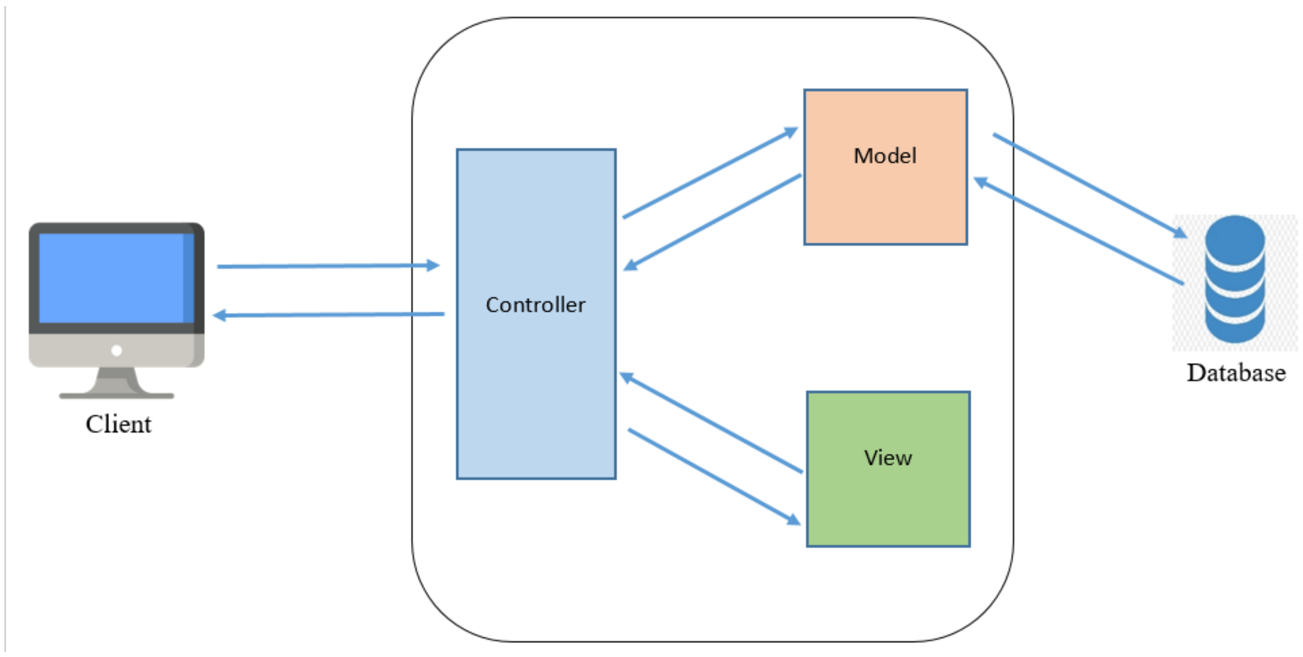
We are going to make a Rest API natours to practice all things we have just learned. We will also dive into what is a rest API, why use a rest API, the difference between dynamic and static website and much more.

MVC architecture

History

Trygve Reenskaug invented MVC. The first reports on MVC were written when he was visiting a scientist at Xerox Palo Alto Research Laboratory (PARC) in 1978/79. At first, MVC was called “Thing Model View Editor” but rapidly changed it to“ Model View Controller”.

The goal of Tygrve was to solve the problem of users controlling a large and complex data set. The practice of MVC has changed over the years. Since the MVC pattern was invented before web browsers, initially was used as an architectural pattern for graphical user interfaces(GUI).



The original MVC

Currently, MVC it's used for designing web applications. Some web frameworks that use MVC concept: Ruby on Rails, Laravel, Zend framework, Cherry, Symphony, etc

MVC Architecture

MVC is an architectural pattern, which means it rules the whole architecture of the applications. Even though often it is known as design pattern, but we may be wrong if we refer it only as a design pattern because design patterns are used to solve a specific technical problem, whereas architecture pattern is used for solving architectural problems, so it affects the entire architecture of our application.

It has three main components:

-Model

-View

-Controller

and each of the has specific responsibilities

MVC Architecture

The main reasons why MVC is used are: First, it doesn't allow us to repeat ourselves and second, it helps to create a solid structure of our web applications.

Model

It is known as the lowest level, which means it is responsible for maintaining data. Handle data logically, so it basically deals with data. The model is actually connected to the database, so anything you do with data. Adding or retrieving data is done in the model component. It responds to the controller requests because the controller never talks to the database by itself. The model talks to the database back and forth, and then it gives the needed data to the controller. Note: the model never communicated with the view directly.

View

Data representation is done by the view component. It actually generates UI or user interface for the user. So at web applications, when you think of the view component, just think the Html/CSS part. Views are created by the data which is collected by the model component, but these data aren't taken directly but through the controller, so the view only speaks to the controller.

Controller

It's known as the main man because the controller is the component that enables the interconnection between the views and the model, so it acts as an intermediary. The controller doesn't have to worry about handling data logic, it just tells the model what to do. After receiving data from the model it processes it , and then it takes all that information it sends it to the view and explains how to represent to the user. Note: Views and models can not talk directly.

Advantage of MVC

- MVC architecture will separate the user interface from business logic and business logic

- Components are reusable.

- Easy o maintain.

- Different components of the application in MVC can be independently deployed and maintained.

- This architecture help to test components independently.

Disadvantages of MVC

-The complexity is high.

-Not suitable for small applications.

-The inefficiency of data access in view.

Rest API

REST is a set of architectural constraints, not a protocol or a standard. API developers can implement REST in a variety of ways.

When a client request is made via a RESTful API, it transfers a representation of the state of the resource to the requester or endpoint. This information, or representation, is delivered in one of several formats via HTTP: JSON (JavaScript Object Notation), HTML, XLT, Python, PHP, or plain text. JSON is the most generally popular file format to use because, despite its name, it's language-agnostic, as well as readable by both humans and machines.

Something else to keep in mind: Headers and parameters are also important in the HTTP methods of a RESTful API HTTP request, as they contain important identifier information as to the request's metadata, authorization, uniform resource identifier (URI), caching, cookies, and more. There are request headers and response headers, each with their own HTTP connection information and status codes.

In order for an API to be considered RESTful, it has to conform to these criteria:

A client-server architecture made up of clients, servers, and resources, with requests managed through HTTP.

Stateless client-server communication, meaning no client information is stored between get requests and each request is separate and unconnected.

Cacheable data that streamlines client-server interactions.

A uniform interface between components so that information is transferred in a standard form. This requires that:

resources requested are identifiable and separate from the representations sent to the client.

resources can be manipulated by the client via the representation they receive because the representation contains enough information to do so.

self-descriptive messages returned to the client have enough information to describe how the client should process it.

hypertext/hypermedia is available, meaning that after accessing a resource the client should be able to use hyperlinks to find all other currently available actions they can take.

A layered system that organizes each type of server (those responsible for security, load-balancing, etc.) involved the retrieval of requested information into hierarchies, invisible to the client.

Code-on-demand (optional): the ability to send executable code from the server to the client when requested, extending client functionality.

Though the REST API has these criteria to conform to, it is still considered easier to use than a prescribed protocol like SOAP (Simple Object Access Protocol), which has specific requirements like XML messaging, and built-in security and transaction compliance that make it slower and heavier.

Documentation

nat

tours

GET get all tours

```
http://localhost:3000/api/v1/tours
```

AUTHORIZATION
Bearer Token

Token

PARAMS

duration[gte]
3

sort
price

BODY raw

```
{
    "user":"aman"
}
```

Example Request get all tours

```
curl --location --request GET 'http://localhost:3000/api/v1/tours' \
--data-raw '{
    "user":"aman"
}'
```

Documentation Settings

POST create a tour

```
http://127.00.0.1/3000api/v1/reviews
```

AUTHORIZATION
Bearer Token

Token

BODY raw

```
{
    "review":"This is an amazing tour",
    "rating":5,
    "tour":"625bcb5f9f06d65c5c9b55a1",
    "user":"6253e7d9a6a79510e445e175"
  }
```

Example Request

create a tour

```
curl --location --request POST 'http://127.00.0.1/3000api/v1/reviews' \
--data-raw '{
    "review":"This is an amazing tour",
    "rating":5,
    "tour":"625bcb5f9f06d65c5c9b55a1",
    "user":"6253e7d9a6a79510e445e175"
  }'
```

GET get id tour

```
http://127.00.0.1/3000api/v1/tours/5
```

BODY raw

```
    "user":"aman"
}
```

Documentation Settings

Example Request

get id tour

```
curl --location --request GET 'http://127.00.0.1/3000api/v1/tours/5' \
--data-raw '{
    "user":"aman"
}'
```

PATCH update a tour

http://127.00.0.1/3000api/v1/tours/

BODY raw

```
{
    "user":"aman"
}
```

Example Request

update a tour

```
curl --location --request PATCH 'http://127.00.0.1/3000api/v1/tours/' \
--data-raw '{
    "user":"aman"
}'
```

DEL delete

http://127.00.0.1/3000api/v1/tours/

BODY raw

```
{
    "user":"aman"
}
```

Documentation Settings

Example Request delete

```
curl --location --request DELETE 'http://127.00.0.1/3000api/v1/tours/' \
--data-raw '{
    "user":"aman"
}'
```

GET top 5 cheapest

http://127.00.0.1/3000api/v1/tours//top-5-cheap

BODY raw

```
{
    "user":"aman"
}
```

Example Request top 5 cheapest

```
curl --location --request GET 'http://127.00.0.1/3000api/v1/tours//top-5-cheap' \
--data-raw '{
    "user":"aman"
}'
```

GET tour stat

http://127.00.0.1/3000api/v1/tours/tour-stats

BODY raw

```
{
    "user":"aman"
}
```

Example Request

tour stat

```
curl --location --request GET 'http://127.00.0.1/3000api/v1/tours/tour-stats' \
--data-raw '{
    "user":"aman"
}'
```

GET monthly plan

http://127.00.0.1/3000api/v1/tours/monthly-plan/:2019

PATH VARIABLES

2019

BODY raw

```
{
    "user":"aman"
}
```

Example Request

monthly plan

```
curl --location --request GET 'http://127.00.0.1/3000api/v1/tours/monthly-plan/:2019' \
--data-raw '{
    "user":"aman"
}'
```

users

Documentation Settings

GET get all users

```
http://localhost:3000/api/v1/users
```

BODY raw

```
{
    "user":"aman"
}
```

Example Request

get all users

```
curl --location --request GET 'http://localhost:3000/api/v1/users' \
--data-raw '{
    "user":"aman"
}'
```

PATCH update user details

```
http://127.00.0.1/3000api/v1/users/updateMe
```

AUTHORIZATION

Bearer Token

Token

BODY raw

```
{

    "name":"new"

}
```

Example Request

update user details

```
curl --location --request PATCH 'http://127.00.0.1/3000api/v1/users/updateMe' \
--data-raw '{

    "name":"new"

}'
```

DEL delete user

http://127.00.0.1/3000api/v1/users/

BODY raw

```
{
    "user":"aman"
}
```

Example Request delete user

```
curl --location --request DELETE 'http://127.00.0.1/3000api/v1/users/' \
--data-raw '{
    "user":"aman"
}'
```

POST update user

http://localhost:3000api/v1/users/6253e7d9a6a79510e445e175

AUTHORIZATION
Bearer Token

Token

BODY raw

```
{
    "email":"amanaman@email.io"
}
```

Example Request update user

```
curl --location --request POST 'http://localhost:3000api/v1/users/6253e7d9a6a79510e445e175' \
--data-raw '{
    "email":"amanaman@email.io"
}'
```

PATCH delete user details

http://127.00.0.1/3000api/v1/users/deleteMe

AUTHORIZATION

Bearer Token

Token

BODY raw

```
{
  "name":"newBring"

}
```

Example Request delete user details

```
curl --location --request PATCH 'http://127.00.0.1/3000api/v1/users/deleteMe' \
--data-raw '{
  "name":"newBring"

}'
```

☰ Documentation Settings ▾

PATCH update user

```
http://127.00.0.1/3000api/v1/users/625cfa6a3d274255346a95a6
```

BODY raw

```
{
    "name":"updated"
}
```

Example Request update user

```
curl --location --request PATCH 'http://127.00.0.1/3000api/v1/users/625cfa6a3d274255346a95a6' \
--data-raw '{
    "name":"updated"
}'
```

GET get me

```
http://127.00.0.1/3000api/v1/users/me
```

AUTHORIZATION
Bearer Token

Token

BODY raw

```
{
  "name":"newBring"

}
```

```
curl --location --request GET 'http://127.00.0.1/3000api/v1/users/me' \
--data-raw '{
  "name":"newBring"

}'
```

Auth

POST sign up

```
http://127.00.0.1/3000api/v1/users/signup
```

BODY raw

```
{
    "name":"aman",
    "email":"aman@email.io",
    "password":"pass@123",
    "passwordConfirm":"pass@123",
    "role":"user"
}
```

Example Request

sign up

```
curl --location --request POST 'http://127.00.0.1/3000api/v1/users/signup' \
--data-raw '{
    "name":"aman",
    "email":"aman@email.io",
    "password":"pass@123",
    "passwordConfirm":"pass@123",
    "role":"user"
}'
```

POST log in

Documentation Settings

http://127.00.0.1/3000api/v1/users/login

BODY raw

```
{
    "email":"admin@email.io",
    "password":"newpass@123"
}
```

Example Request

log in

```
curl --location --request POST 'http://127.00.0.1/3000api/v1/users/login' \
--data-raw '{
    "email":"admin@email.io",
    "password":"newpass@123"
}'
```

POST forget password

http://127.00.0.1/3000api/v1/users/forgotPassword

AUTHORIZATION

Bearer Token

Token

BODY raw

```
{
    "email":"bring@email.io"


}
```

Example Request

forget password

Documentation Settings

```
curl --location --request POST 'http://127.00.0.1/3000api/v1/users/forgotPassword' \
--data-raw '{
    "email":"bring@email.io"


}'
```

PATCH reset password

http://127.0.0.1:3000/api/v1/users/resetPassword/4a63cd54d3a0097c3d37b4b5e42a4635cde48ff0f5368b07311802d9209c1fe9

AUTHORIZATION
Bearer Token

Token

BODY raw

```
{
    "password":"newpass@123",
    "passwordConfirm":"newpass@123"

}
```

Example Request

reset password

```
curl --location --request PATCH 'http://127.0.0.1:3000/api/v1/users/resetPassword/4a63cd54d3a0097c3d37b4b
--data-raw '{
    "password":"newpass@123",
    "passwordConfirm":"newpass@123"

}'
```

PATCH update password

Documentation Settings

```
http://127.00.0.1/3000api/v1/users/updateMyPassword
```

AUTHORIZATION
Bearer Token

Token

BODY raw

```
{

    "passwordCurrent":"pass@123",
    "password":"newpass@123",
    "passwordConfirm":"newpass@123"

}
```

Example Request

update password

```
curl --location --request PATCH 'http://127.00.0.1/3000api/v1/users/updateMyPassword' \
--data-raw '{

    "passwordCurrent":"pass@123",
    "password":"newpass@123",
    "passwordConfirm":"newpass@123"

}'
```

reviews

POST create a review

```
http://127.00.0.1/3000api/v1/reviews
```

AUTHORIZATION

Bearer Token

Documentation Settings

Token

BODY raw

```
{
    "review":"This is an amazing tour",
    "rating":5,
    "tour":"625bcb5f9f06d65c5c9b55a1",
    "user":"6253e7d9a6a79510e445e175"
  }
```

Example Request

create a review

```
curl --location --request POST 'http://127.00.0.1/3000api/v1/reviews' \
--data-raw '{
    "review":"This is an amazing tour",
    "rating":5,
    "tour":"625bcb5f9f06d65c5c9b55a1",
    "user":"6253e7d9a6a79510e445e175"
  }'
```

GET get a review for a specific tour

```
http://127.00.0.1/3000api/v1/tours/625bcb5f9f06d65c5c9b55a1/reviews
```

AUTHORIZATION

Bearer Token

Token

Example Request

get a review for a specific tour

```
curl --location --request GET 'http://127.00.0.1/3000api/v1/tours/625bcb5f9f06d65c5c9b55a1/reviews'
```

Documentation Settings

DEL delete review

http://127.00.0.1/3000api/v1/reviews/625d25dd8c49007338c6c872

AUTHORIZATION

Bearer Token

Token

BODY raw

```
{
    "review":"This is an amazing tour",
    "rating":5,
    "tour":"625bcb5f9f06d65c5c9b55a1",
    "user":"6253e7d9a6a79510e445e175"
  }
```

Example Request

delete review

```
curl --location --request DELETE 'http://127.00.0.1/3000api/v1/reviews/625d25dd8c49007338c6c872' \
--data-raw '{
    "review":"This is an amazing tour",
    "rating":5,
    "tour":"625bcb5f9f06d65c5c9b55a1",
    "user":"6253e7d9a6a79510e445e175"
  }'
```

PATCH update a review

http://127.00.0.1/3000api/v1/reviews

AUTHORIZATION

Bearer Token

Token

BODY raw

```
{
    "review":"This is an amazing tour",
    "rating":5,
    "tour":"625bcb5f9f06d65c5c9b55a1",
    "user":"6253e7d9a6a79510e445e175"
  }
```

Example Request — update a review

```
curl --location --request PATCH 'http://127.00.0.1/3000api/v1/reviews' \
--data-raw '{
    "review":"This is an amazing tour",
    "rating":5,
    "tour":"625bcb5f9f06d65c5c9b55a1",
    "user":"6253e7d9a6a79510e445e175"
  }'
```

GET get one review

```
http://127.00.0.1/3000api/v1/reviews/625d26b99f45dd7394eacda4
```

AUTHORIZATION

Bearer Token

Token

BODY raw

```
{

    "rating":3

  }
```

Example Request — get one review

```
curl --location --request GET 'http://127.00.0.1/3000api/v1/reviews/625d26b99f45dd7394eacda4' \
--data-raw '{

    "rating":3

  }'
```

GET get all reviews

http://127.00.0.1/3000api/v1/reviews

AUTHORIZATION

Bearer Token

Token

BODY raw

```
{
    "review":"This is an amazing tour",
    "rating":5,
    "tour":"625bcb5f9f06d65c5c9b55a1",
    "user":"6253e7d9a6a79510e445e175"
  }
```

Example Request

get all reviews

```
curl --location --request GET 'http://127.00.0.1/3000api/v1/reviews' \
--data-raw '{
    "review":"This is an amazing tour",
    "rating":5,
    "tour":"625bcb5f9f06d65c5c9b55a1",
    "user":"6253e7d9a6a79510e445e175"
  }'
```

POST create a review for a specific tour

Documentation Settings

```
http://127.00.0.1/3000api/v1/tours/625bcb5f9f06d65c5c9b55a1/reviews
```

AUTHORIZATION

Bearer Token

Token

BODY raw

```
{
    "review":"This is an amazing tour",
    "rating":5,
    "tour":"625bcb5f9f06d65c5c9b55a1",
    "user":"6253e7d9a6a79510e445e175"
  }
```

Example Request

create a review for a specific tour

```
curl --location --request POST 'http://127.00.0.1/3000api/v1/tours/625bcb5f9f06d65c5c9b55a1/reviews' \
--data-raw '{
    "review":"This is an amazing tour",
    "rating":5,
    "tour":"625bcb5f9f06d65c5c9b55a1",
    "user":"6253e7d9a6a79510e445e175"
```

Authorization

Authorization is the process of allowing authenticated users access to resources by determining whether they have system access permissions. By giving or denying specific licences to an authenticated user, authorization enables you to control access privileges.

So, authorization occurs after the system authenticates your identity, granting you complete access to resources such as information, files, databases, funds, places, and anything else. That said, authorization affects your capacity to access the system and the extent to which you can do so.

What is JWT?

JSON Web Tokens (JWT) are an RFC 7519 open industry standard for representing claims between two parties. For example, you can use jwt.io to decode, verify, and produce JWT.

JWT specifies a compact and self-contained method for communicating information as a JSON object between two parties. Because it is signed, this information can be checked and trusted. JWTs can be signed using a secret (using the HMAC algorithm) or an RSA or ECDSA public/private key combination. In a moment, we'll see some examples of how to use them.

Login and reset password

To summarize, I present to you a possible implementation of the Reset Password Flow using JWT:

User requests for password reset

UI sends POST request to servers to generate a JWT in the form of a link sent to the user through email

— The JWT payload consists of the username to uniquely identify the user. JWT expiration is set to a limited time, say 30 mins.

— The JWT signature is signed with a secret: the user's password hash (not known to the public)

— The JWT could be appended in the query of the link: https://exampletest.com/reset/password?token={Insert JWT here}

The user clicks on the reset password link and redirected to a page

The front-end sends the JWT parsed from the query string to the backend to Verify the JWT using the user's password hash (the user is identified using the username in the JWT payload).

If there is no error, the user is presented with a Reset Password Form. Else, an error should be shown.

Unit testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. This testing methodology is done during the development process by the software developers and sometimes QA staff. The main objective of unit testing is to isolate written code to test and determine if it works as intended.

Unit testing is an important step in the development process, because if done correctly, it can help detect early flaws in code which may be more difficult to find in later testing stages.

Unit testing is a component of test-driven development (TDD), a pragmatic methodology that takes a meticulous approach to building a product by means of continual testing and revision. This testing method is also the first level of software testing, which is performed before other testing methods such as integration testing. Unit tests are typically isolated to ensure a unit does not rely on any external code or functions. Testing can be done manually but is often automated.

Unit testing with JS

JavaScript Unit Testing is a method where JavaScript test code is written for a web page or web application module. It is then combined with HTML as an inline event handler and executed in the browser to test if all functionalities are working as desired. These unit tests are then organized in the test suite.

How to unit test rest API

When building software, we often forget the importance of testing. Testing not only ensures your application or system is working as expected, but it also helps manage new changes in specification or implementation.

We can’t know whether the system is working as it should be unless there’s a mechanism in place to check if the system works well after new changes.

There are different types of testing for different test approaches. However, the most popular and important is unit testing. Unit testing is basically testing if a unit or component of the system is working as expected. You either just call the component, if no input is required, or give it an input and determine the output.

In the context of REST API, a unit is a single endpoint request, and writing a unit test for this particular API depends on what you want to test in its response base on the request sent.

In a single API endpoint request, you can test its response for the combination of:

Response body

Response header (authorization)

Response status code

With this, you’re able to assert the expected response body, header, and status code. This process requires an HTTP request client library, an assertion library, a testing framework and a bit of coding to get the optimal output.

Tools for unit testing in JS

Mocha

Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple and fun. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases. Hosted on GitHub.

Chai

Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any JavaScript testing framework.

Chai http

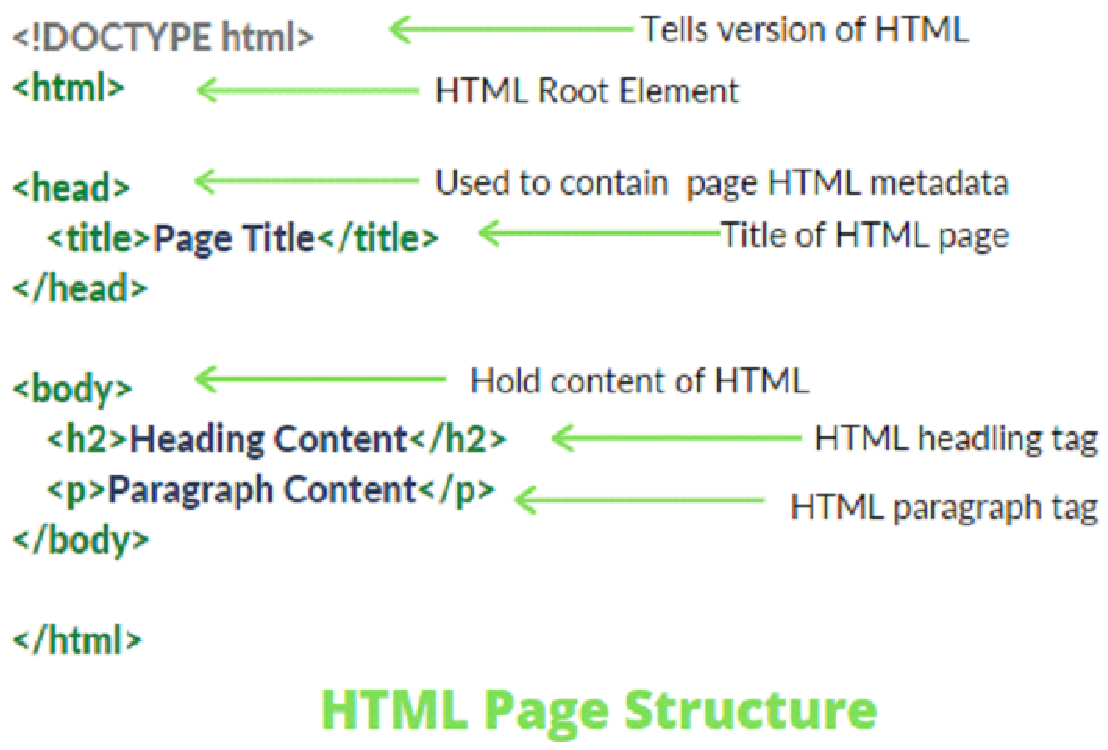
Chai HTTP provides an interface for live integration testing via super agent. To do this, you must first construct a request to an application or URL.

Chapter 4- Front End

HTML

HTML stands for HyperText Markup Language. It's a markup language that's used to create web pages. HTML is a markup language that combines hypertext with markup. The link between web pages is defined as hypertext. The text document within the tag that defines the structure of web pages is defined using a markup language. This language is used to annotate (add notes to) text so that a computer can understand it and manipulate it appropriately. Human-readable markup languages include HTML and others. Tags are used in the language to specify what type of text processing is required.

HTML is a markup language that the browser uses to transform text, pictures, and other content so that it can be displayed in the desired format. Tim Berners-Lee invented HTML in 1991. HTML 1.0 was the first version, but HTML 2.0 was the first standard version, released in 1995.



```
<!DOCTYPE html>
<html>

<head>
  <title>Page Title</title>
</head>

<body>
  <h2>Heading Content</h2>
  <p>Paragraph Content</p>
</body>

</html>
```

Figure 46: HTML Page Structure

CSS- Cascading Style Sheets

CSS (Cascading Style Sheets) is a stylesheet language for describing the presentation of an HTML or XML document (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should look on a screen, on paper, in speech, or in other media.

CSS is a basic language for the open web that is supported by all major browsers. Separate parts of the CSS specification were previously developed in parallel, allowing for the versioning of the most recent suggestions.

CSS is not a programming language like HTML. It's also not a markup language. CSS is basically a style sheet language. CSS is used to style HTML components selectively.

Flex layout :

Flex Layout is included to give the component a neat appearance. The Flexbox is used to specify the children of a component layout. We can archive the correct layout by using the flexDirection, justifyContent, and alignItems attributes.

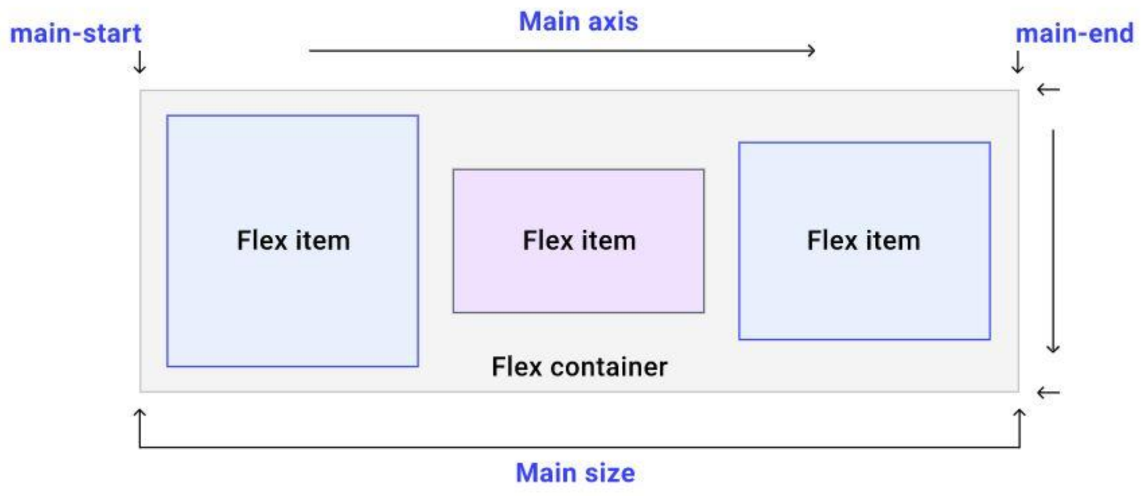


Figure 47: FlexBox Structure

JavaScript

JavaScript is a lightweight, cross-platform object-based scripting language. JavaScript is a translated language, not a compiled language. The JavaScript Translator (which is built within the browser) is in charge of translating JavaScript code for web browsers.

JavaScript (js) is a lightweight object-oriented programming language that is used to script web pages by various websites. When applied to an HTML document, it is an interpreted, full-featured programming language that enables dynamic interactivity on websites. It was first released in 1995 to allow users to add programmes to web pages in the Netscape Navigator browser. Since then, all other graphical web browsers have embraced it. Users can utilize JavaScript to create modern web applications that interact immediately without having to reload the page every time. Js is used on a typical website to promote interactivity and simplicity.

There is no connection between JavaScript and the Java programming language. When Java began gaining prominence in the market, the name was suggested and provided. Databases like CouchDB and MongoDB use JavaScript as their scripting and query language, in addition to web browsers.

- Synchronous JavaScript: Synchronous refers to the execution of each statement of code in a sequential order. So, in essence, a statement must wait for the execution of the previous statement.

- Asynchronous JavaScript: Asynchronous code permits the program to be executed immediately, whereas synchronous code prevents the remaining code from being executed until the current one is completed. This may not appear to be a major issue, but when viewed in context, it can result in the User Interface being delayed.

Typescript

JavaScript was introduced as a client-side programming language. JavaScript has emerged as an emergent server-side technology as a result of the creation of Node.js. However, as JavaScript code develops in complexity, it becomes more difficult to maintain and reuse. Furthermore, JavaScript's failure to embrace Object Orientation, rigorous type checking, and compile-time error checks hinders it from prospering as a full-fledged server-side technology in the enterprise. To fill this void, TypeScript was introduced.

TypeScript is JavaScript for application-scale development.

TypeScript is a compiled, strongly typed, object-oriented language. TypeScript is a set of tools as well as a programming language. JavaScript is compiled to TypeScript, which is a typed superset of JavaScript. To put it another way, TypeScript is JavaScript with a few more features.

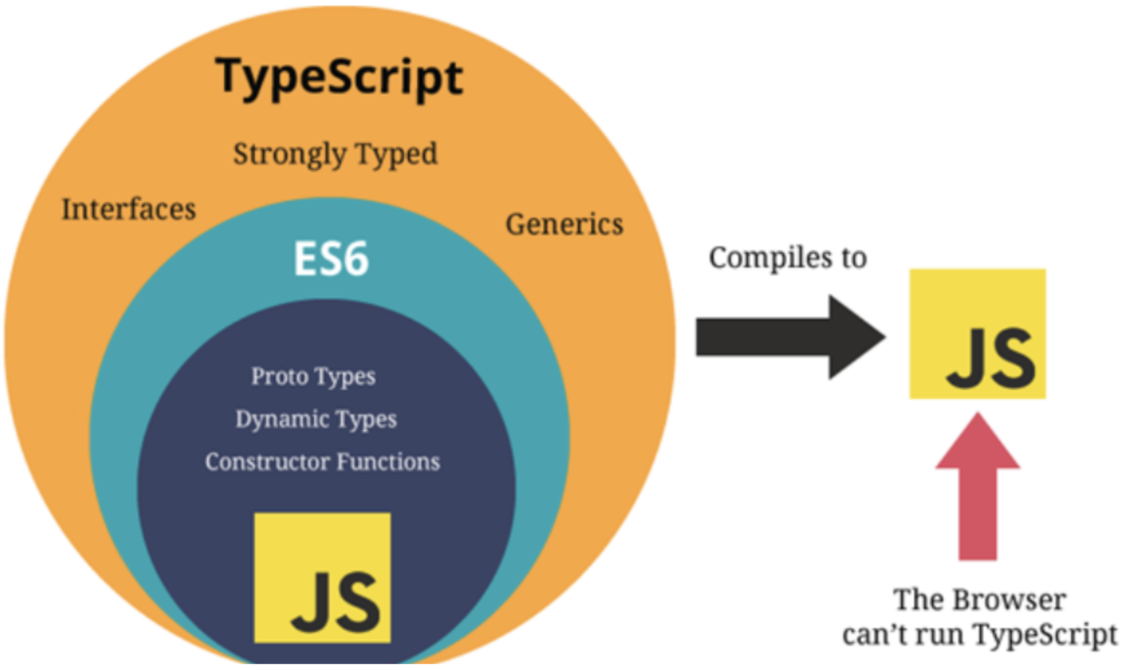


Figure 48: Compilation process of TypeScript

Typescript is the outer domain of JavaScript. JavaScript is the beginning and the end of TypeScript. The essential building parts of your application are taken from JavaScript by

Typescript. To utilize TypeScript, you simply need to know JavaScript. For the purpose of execution, all TypeScript code is translated to its JavaScript equivalent.

JS libraries are supported by TypeScript. Any JavaScript code can consume compiled TypeScript. All existing JavaScript frameworks, tools, and libraries can be used with TypeScript-generated JavaScript.

JavaScript is TypeScript. JavaScript files can be converted to TypeScript just by renaming the .js file to .ts and compiled with other TypeScript files.

TypeScript is portable. TypeScript is cross-browser, device, and operating system compatible. It can run in any JavaScript-enabled environment. TypeScript does not require a separate virtual machine or a specific runtime environment to run, unlike its competitors.

Advantages of TypeScript

- TypeScript always highlights compilation mistakes during development (pre-compilation). Because JavaScript is an interpreted language, it is less likely to encounter runtime problems.
- Static/strong typing is supported in TypeScript. This means that type correctness can be verified during the compilation process. JavaScript does not support this feature.
- TypeScript is nothing more than JavaScript with some ES6 capabilities thrown in.

Chapter 5- Result and Conclusion

This internship was indeed a pool of knowledge, not only have I gained

knowledge in Full Stack Development but I have also learned about how development of any project takes place, how team works, how the work of each employee is tracked, how work is distributed between different team mates, what are the different stages of development, what are the technical problems that one faces in the development of any project, what all things are required before the development of any project, what the code base should be like and what norms need to be followed in the development. I have learned how to write code in such a manner that its understood by every team member of my team. I have also gained some knowledge on being an active team member and leading my fellow team members to great ideas.